7. Kristina

Program Name: Kristina.java **Input File: kristina.dat**

Kristina has been studying prefix and postfix notation for arithmetic expressions and has asked you to create a combination prefix/postfix calculator program that will check her manual evaluation of expressions of those forms.

Prefix notation places operators before their operands and operations have to be delayed until two operands are available. Consider the prefix example: **+ - * 7 5 - 9 6 -5**

*** 7 5** is 7 * 5 = 35
- 9 6 is 9 − 6 = 3
- 35 3 is 35 − 3 = 32
+ 32 -5 is 32 + -5 = 27

Postfix places operators after their operands which means the needed operands are available as soon as an operator is encountered. Consider the postfix example: **7 5 * 9 6 - - -5 +**

7 5 * is 7 * 5 = 35
9 6 - is 9 − 6 = 3
35 3 - is 35 − 3 = 32
32 -5 + is 32 + -5 = 27

Kristina wants to work with only integers and wants to use **^** as an exponent operator. For example, 2 **^** 5 is 32.

Can you create the program for Kristina?

Input: First line of data file contains a positive integer T, the number of test cases that follow with $1 \le T \le 25$. The following T lines will start with either "PRE" for a prefix expression or "POST" for a postfix expression. Those will be followed by a single space and a properly formed expression of the indicated notation. The combination of operators (^, *, /, +, and −) and integers, N, will be separated by single spaces with $-100 \le N \le 100$. However, exponent and other operations are guaranteed to not result in values that exceed a standard Java integer. Expressions are also guaranteed to not result in division by 0. Length of lines will not exceed 200 characters.

Output: For each test case, display the resulting integer value.

Sample input:

```
6
PRE + - * 7 5 - 9 6 -5
POST 7 5 * 9 6 - - -5 +
POST 2 8 3 4 2 * / + -
PRE - 2 + 8 / 3 * 4 2
POST -39 9 + -35 -44 * -100 / -
PRE + + -97 - * 47 29 / ^ 3 5 100 ^ 2 10
```

Sample output:

```
27
27
-6
-6
-15
2288
```